TURING 图灵程序设计丛书 Web开发系列

Apress®

HTML Mastery

Semantics, Standards, and Styling

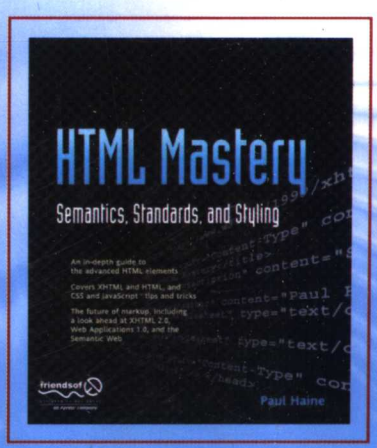
精通HTML

语义、标准和样式

[英] Paul Haine 著

杨明军 等译

- 畅销书《精通CSS》姊妹篇
- 透彻剖析HTML本质
- 涵盖微格式、语义网、XHTML 2.0和HTML 5等新技术



人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 Web开发系列

精通 HTML
语义、标准和样式

HTML Mastery
Semantics, Standards, and Styling

[英] Paul Haine 著
杨明军 等译

人民邮电出版社
北 京

图书在版编目（CIP）数据

精通 HTML：语义、标准和样式 /（英）波恩（Haine, P.）著；杨明军等译. —北京：人民邮电出版社，2008.2
（图灵程序设计丛书）
ISBN 978-7-115-17090-3

I. 精… II. ①海…②杨… III. 超文本标记语言，HTML-程序设计 IV. TP312

中国版本图书馆 CIP 数据核字（2007）第 169742 号

内 容 提 要

本书深入地探讨了 (X)HTML 及相关技术包括 CSS、微格式、语义网等，重点阐述了如何在恰当的时候使用恰当的标签，全书始终贯彻现代的 Web 设计理念，从而使读者可以学习如何充分利用各种标记提供的多样性，创建语义丰富和结构合理的网站。

本书适合具备初步 HTML 和 CSS 知识的 Web 设计开发人员阅读。

图灵程序设计丛书

精通 HTML：语义、标准和样式

◆ 著 [英] Paul Haine
译 杨明军 等
责任编辑 陈兴璐

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
三河市海波印务有限公司印刷
新华书店总店北京发行所经销

◆ 开本：800×1000 1/16
印张：12
字数：284 千字 2008 年 2 月第 1 版
印数：1 – 5 000 册 2008 年 2 月河北第 1 次印刷

著作权合同登记号 图字：01-2007-4264 号

ISBN 978-7-115-17090-3/TP

定价：35.00 元

读者服务热线：(010)88593802 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

译 者 序

如果说因特网改变了人类的生活方式，我想没有人会反对，而且它还正在更加深刻地改变着我们。其中 Web 扮演的角色至关重要。通过全球通用的规范，Web 将世界上无数的文档彼此链接起来，实现了信息的共享，方便了资料的发布和查询。随着技术的进步以及人类对信息技术需求的增加，Web 技术也发生了深刻的变化。人们不再满足于文档的共享，而是希望 Web 能够完成诸如电子商务、人际沟通之类的更多业务。Web 深入人类生活，基于“信息面前，人人平等”的诉求，要求它必须满足范围越来越广的用户需求，比如近视患者和盲人。此外，随着因特网的“爆炸式”发展，Web 上的数据每时每刻都在快速增长，人类需要借助更多的“代理”才能够方便地聚合、充分利用这些信息。作为 Web 信息的承载者，网页必须能够满足以上要求。将内容、结构和表示三者分离是一种切实可行的方案，按照这种方式构建的页面既能满足普通用户在美感、可用性等方面的要求，又能方便搜索引擎、语音设备等用户代理的浏览。

多样性是 Web 世界的主流，没有任何一款产品能够独领风骚，IE 也不例外。对于 Firefox、Opera 以及 Safari 等浏览器来说，新兴市场充满机遇。而对于 Web 设计师来说，只有编写符合 Web 标准的页面，才能够在尽可能多的用户代理上取得理想的浏览效果。本书详细地讲解了常用的 HTML 页面元素，为 Web 设计师日常的设计和编码工作提供了详细的指南。同时本书还介绍了 Web 技术发展趋势，可以为读者未来的学习和研究指明方向。

本书是畅销书《精通 CSS》的姊妹篇，其风格和层次都相同。HTML 本身是一种简单的技术，掌握它并不难，关键在于如何用好并精通。其核心问题是在什么情况下用什么标签最合适。本书正是为了解决这个问题而写，通过本书的学习，读者能够在 Web 设计方面更上一层楼。市面上（包括国外）这种中高级层次的网页设计书非常少，因而本书非常值得一看。

本书主要由杨明军翻译。参与翻译的还有马蓉、焦贤龙、张杰良、肖枫涛、刘齐军、韩智文、张聪、闫志强、唐玲艳、肖国尊等。Be Flying 工作室负责人肖国尊负责本书译员的选定、翻译质量和进度的控制与管理。敬请广大读者提供反馈意见，读者可以将意见发邮件至 be-flying@sohu.com，我们会仔细查阅读者发来的每一封邮件，以求进一步提高今后译著的质量。同时欢迎各位进入 Be Flying 工作室博客 http://blog.csdn.net/be_flying/，或者 China-Pub 互动网上的宣传链接 http://www.china-pub.com/main/sale/renwu/GetInfo.asp?theID=64，来了解 Be Flying 工作室的所有其他译著。

前　言

最开始的时候，HTML 诞生了，它表现得非常出色；随着时间的推移，我们拥有了各种各样的 HTML，但它们好像又不尽如人意了；再后来我们开始使用 XHTML，它表现得更好，虽然很多时候它还没有发挥真正的潜力。

就在几年之前，Web 设计师还无需理解 HTML 或者 CSS，即使需要理解也不必精通，只需了解基本的相关知识就足够了。相对地，精通 Photoshop 和 Dreamweaver 等软件则可能更重要一些。那时可以直接利用图片生成一个网站，不会考虑其背后的标记及其状态——不必管它的设计是精良还是粗糙，也不必管它是否高效、是否有意义。不过，几年以前也没有什么选择余地。你只能使用表格和间隔图片来布局一个网站而避免使用那些语义标记，因为当时的浏览器中并不支持 Web 标准。

这种设计的后果就是网站的负载往往很重，也经常会因此而变得非常慢，并且通常只能在一种浏览器中正常工作，这样的网站也难以更新和维护。为了能够打印页面，还需要将同样的内容用两个不同的页面表示，这不利于搜索引擎创建索引、理解内容和进行排名。于是又导致人们使用大量的搜索优化技巧、堆积着大量关键字的<meta>元素以及针对每个搜索引擎的入口页面等。另外，表示（界面外观）和行为（通常是 JavaScript）也都混合着内容，页面毫无意义且没有逻辑结构——那时人们考虑的是页面的外观而不是其意义。

在当时，Web 设计师的工作不那么好做。

如今，新生派的 Web 设计师就需要学习更多有关基础结构的知识。他们需要知道如何编写(X)HTML、如何编写 CSS，如何解决某个布局 bug 在 3 个不同版本的 IE 以及在 FireFox、Opera 和 Safari 中引起的不同问题（如果要求更高一点，还需要懂得如何在设计最开始的时候就避免这些布局 bug）。Web 设计师再次开始学习如何手工地编写(X)HTML 代码，但是，当习惯了在 Dreamweaver 的设计视图中构建基于表格的网站之后，突然转而在 Dreamweaver 的代码视图中手工编写网站的(X)HTML 代码，你往往会遇到很多困难。

本书主要面向满足以下条件的 Web 设计师：你可能已经学习了创建一个两栏布局所需要的(X)HTML 和 CSS 知识；也可能曾经花费很多时间使用 FrontPage 或 Dreamweaver，而现在希望学习更多用于构建自己网站的底层技术；也可能想要进一步提升自己使用标记的水平。那么就从本书开始你的深入学习之旅吧！本书不是从最基本的知识开始讲授(X)HTML，因为我们假设你

已经拥有了相关的基本知识；当然本书也并不会关注于使用 CSS 来设计整个网站，尽管书中也会包含几个示例，采用 CSS 和 JavaScript 美化你新编写的、基于标准的标记。

本书的目的是更深入地探索(X)HTML，从而学习如何充分利用各种标记提供的多样性，来帮助你创建语义丰富和结构合理的网站。按照这种方式构建的网站能让你、网站访问者、检索网站的搜索引擎都理解。通过本书的学习，你将会学到如何使用短语元素来改善页面文本，明智而又灵活地利用表示性元素来创建内容翔实而又有用的表格和表单，以及找到更好的增强网页内容的方法，而不只是简单地点击网页设计编辑器的倾斜（I）或加粗（B）按钮。

本书约定

当书中出现 HTML 或者 XHTML 的时候，分别指的是 HTML 4.01 和 XHTML 1.0。如果论述的内容同时适用于二者，将写做(X)HTML。

书中出现的“现代浏览器”指的是与标准兼容（或者差不多能够兼容）的浏览器。在写作本书的时候，这些浏览器包括 Opera、Firefox（以及其他浏览器，比如 Camino 或者 Mozilla，这几款浏览器都使用了相同的渲染引擎）、Safari 和 IE 7[①]。假定随着新版标准的发行，这些现代浏览器将会继续与标准兼容。

重要的文字和概念会在第一次出现时用楷体突出显示。代码采用代码体，若代码在一行中不能完整显示，会使用箭头➥，如下所示：

```
This is a very, very long section of code that should be written all ➥
on the same line without a break.
```

致谢

感谢在写作本书的这 8 个月里面，每个包容我的人：Vikki、Emma、Thom、Verity、我的父母、整个 Britpack 小组，还有很多其他人，这里就不一一列举了。感谢参与本书的 Apress 和 friends of ED 出版社的每位员工，感谢 Chris Mills 将这个项目摆在了最重要的位置上，感谢 Ian Lloyd[②] 为本书做了技术审稿。

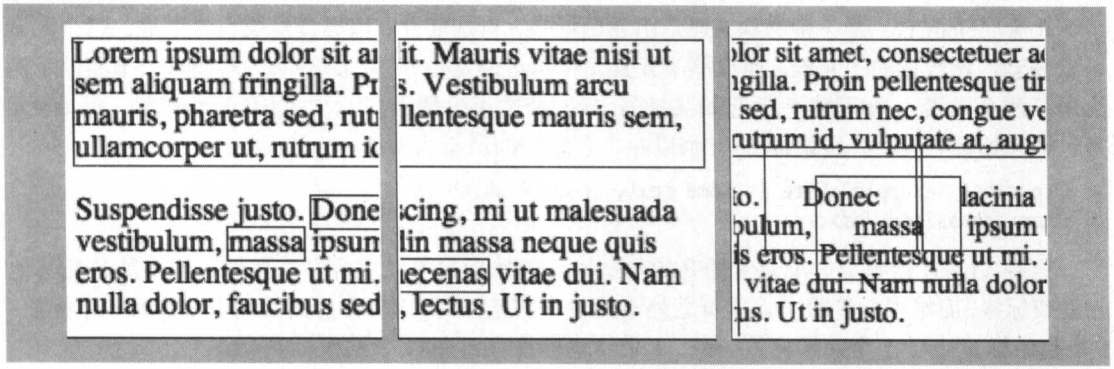
特别感谢 Leon、Ian、Helen 还有 gv，感谢你们在我忙于写作期间能够让我的网站稳定运行。

① IE 7 是临时包含进来的，在本书写作的时候，其最终版才刚刚公开发行。尽管它对标准的兼容性有所增强，但是似乎还没有赶上 Opera 和 Firefox 的水平。

② 本书的技术审稿人，也是一位优秀的 Web 设计师，Web 标准项目成员。——编者注

目　录

第1章 开 始



精通 HTML 并不仅仅是了解有哪些标签可用、其具体意义是什么，了解 HTML 本身也同样重要——也就是说，要理解什么是标签、什么是属性以及如何使用它们，掌握 HTML 和 XHTML 之间的区别，了解什么是 doctype（文档类型）以及如何读取它等。了解 HTML 不仅可以帮助你深入地理解它，还可以在大家一起讨论的时候帮助别人更好地理解你的意图。

本章主要包括 3 部分内容。第一部分主要介绍在讨论和编写 HTML 时要用到的术语；第二部分比较作为同一语言两个版本的 HTML 和 XHTML 之间的不同之处，研究一些有关它们的常见误解；在第三部分我们将一个典型的 HTML 和 XHTML 文档拆解开来，逐一分析其含义和作用。

如果你对这些主题已经很熟悉，那么可以直接跳到下一章。但是，我仍然建议你像初学者那样仔细地阅读本章——这不会占用你很长时间，却又绝对是值得的，因为它包含了很多有用信息。此外，如果你能够比同行了解更多的 HTML 知识，那岂不是既时尚又超酷，有谁不想呢？

1.1 (X)HTML 术语

如果你想制作专业的(X)HTML文档，给你的朋友和同事留下深刻的印象，那么只做不说可不行。使用正确的术语对于避免混淆、帮助你本人及他人加深理解都极其重要。举例来说，假如有人提到“title标签（标题标签）”，那么他指的是显示在浏览器标题栏中的文档标题呢，还是当光标停留在一个元素（通常是一幅图片或一个链接）上面时显示出来的提示信息（title属性）

呢？此外，他所指的也有可能是出现在页面中的文本标题（heading），而此时很有可能是某一个<h1>元素中的内容。注意，这里有标签、元素和属性，而它们是完全不同的。

为了保证大家在进行后续学习之前能够在同一水平线上，我将会在本节中解释在讨论(X)HTML时经常会遇到的每一个术语的具体含义。此外，我还会讨论其他一些有可能造成混淆的常用术语，其中包括部分（div）、跨度（span）、标识（id）、类别（class）、块（block）和行内（inline）。

1.1.1　元素和标签

元素（element）是一种结构，通常由起始标签、可选属性、内容及结束标签组成。每个元素可以嵌套任意数量的子元素，而这些子元素同样也可能是由标签、属性和内容组成。下面的示例给出了两个元素： 元素包括起始尖括号（<）和结束尖括号（>）之间的任何内容，而元素包括起始标签、结束标签以及二者之间的所有内容。

```
<p class="example">Here is some text, some of which is
<em>emphasized</em></p>
```

标签（tag）标明了一个元素的开始与结束。起始标签可以包含多个属性，但不能包括其他元素或标签，而结束标签除了它本身之外不能包含其他任何内容。在前面的例子中共有4个标签：1个起始标签 、1个起始标签、1个结束标签和1个结束标签 。

并不是所有的元素都有结束标签。比如，<img>、
、<meta>与<hr>都可以看作是自结束元素（self-closing element）、空元素（empty element）或置换元素（replaced element）。这些元素都不是容器标签，即不能写成这样：“<hr>内容</hr>”或者“
内容</br>”，而且任何内容或格式化[①]均作为属性值处理（如果需要更多信息，请参见下一节）。在HTML中，通常将自结束元素简单地写成<img>、
、<meta>或<hr>。而在XHTML中，要求自结束元素中有一个空格后面跟着一个斜杠，例如<img />、
、<meta />或<hr />。

留意<script>元素：作为一个容器，它必须有结束标签，尽管它可以保持内容为空而使用src属性来引用外部脚本。这个问题实际上更加复杂，因为Opera（版本9和更高版本）和Safari都支持自结束的<script>，所以此时该元素可以正常工作，但其他浏览器中仍然不支持这种用法，所以这种用法在这些浏览器中将是不合法的。

1.1.2　属性

属性（attribute）出现在标签之中，它们仅仅可以包含该属性的值，例如：

```
<p class="example">Here is some text, some of which is
<em>emphasized</em></p>
```

① 除了CSS格式之外。

这个例子展示了类别（class）属性。一个属性可包括多个由空格隔开的值，如果你希望将多个类别应用到同一个属性，这就非常有用。例如，有两个样式，一个命名为 example，另一个命名为 reference，可参考下面的做法将其应用到同一个段落：

```
<p class="example reference">
```

此外，你以前可能已经遇到过包括 alt、src 和 title 在内的其他一些属性。此外还有更多的属性，其中一些是与元素相关的（如<option>标签的 selected 属性），而另外一些则不是（如 class 和 id 属性）。另外，我希望大家在学习完本书之后不再使用 alt 标签，千脆忘记它吧！

1.1.3 应该了解的其他术语

在描述过诸如元素、标签、属性等术语之后，下面我们会转而介绍另外的几个术语，这些术语在编写(X)HTML 文档时也是必须了解的：部分（div）、跨度（span）、标识（id）、类别（class）、块（block）和行内（inline）。如同元素、标签和属性一样，作为一名 Web 设计师，也会经常遇见这些术语，并且很好地理解它们是什么以及如何起作用也同样重要。

人们经常会将这些术语混为一谈，因为他们误解了这些术语的目的，或者在它们之间进行错误的关联。（比如，将 id 属性只与 标签关联，而将 class 属性与标签关联。）

1．部分和跨度

如果用得好的话，部分和跨度可以使页面的结构具有逻辑性，并且还可以为你留置一个钩子，当需要的时候用它们来应用任何 CSS 或者 DOM 脚本。但如果用得不好，这两个标签会使文档变得凌乱不堪，标记、样式和脚本也会变得复杂。我们将会在第 6 章深入地讲解这两个标签，而在此处仅仅是简单地给出二者之间的主要区别。

部分（div，“division”的简写）用于划出一块内容，例如文档的主内容块、主导航栏、页眉、页脚。因此，我们可以将其看作是一个块（block）元素。它可以嵌套子元素，如果需要还可以包含更多的 div 元素，但是不能将其放置在行内元素里面。举例来说，一个简单的网站可能包含一个页眉、一个主内容栏、一个次要内容栏和一个页脚。用(X)HTML 编写如下所示：

```
<div id="header">
    ...
</div>
<div id="mainContent">
    ...
</div>
<div id="secondaryContent">
    ...
</div>
<div id="footer">
    ...
</div>
```

然后，这些内容块能按照 CSS 样式的要求来进行放置和显示。

跨度（span）用于在一个块元素内部划分出区域，而有时也在另一个行内元素内部划分出区域。它是一个行内（inline）元素，在这一点上与、或<a>都是一样的，只不过它没有任何语义——它只是一个通用容器。它本身能进一步包含其他行内元素，例如包含更多的 span 元素。举例来说，要将某一段的段首两个单词设置为红色而其余文字保持黑色不变，则可以使用实现这种效果：

```
<p><span class="leadingWords">The first</span> two words of this ➥
paragraph can now be styled differently.</p>
```

span 元素不能包含块元素——也就是说，不能在中放置 并期待它能按照预期的方式工作。

部分和跨度广泛地应用于微格式（microformat），我将在第 5 章中对此问题进行讲解。

2．块元素和行内元素

为了稍微地简化，(X)HTML 中的每个元素都包含在一个“盒子”中，这个盒子要么是一个块级元素，要么是一个行内级元素。通过在 CSS 中添加边框或者轮廓，就可以看到“盒子”的存在。图 1-1 中的示例给出了盒子应用于块和行内的差别。

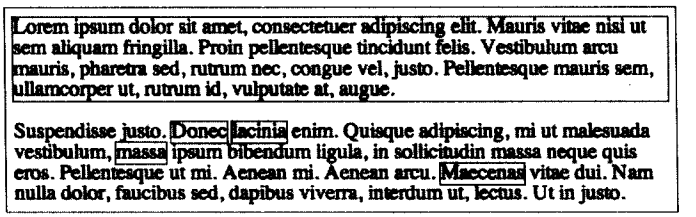


图 1-1　“盒子”模型，应用到块元素和行内元素

块级元素（block-level box，比如部分、段落或者标题均属此类），在渲染的时候需要在文档中另起一行，并且强制后面的元素在该元素下方另起一行开始渲染。这就意味着在没有样式的文档中，块级元素都是垂直堆叠在一起的，并沿所包含元素的左边排列。它们还扩展填满所包含元素的宽度。如果不使用 CSS 的话，就不可能将两个块元素并列放置在一起。

无论将行内级元素（inline-level box，比如或者均属此类）放置到文档中的哪个地方，这些元素都能够渲染，并且不会强制要求换行。除非在 CSS 中明确地要求行内级元素垂直排列或者用一个新的块级元素将它们隔开，否则这些行内级元素都将沿着水平方向排列，而不是垂直排列。行内级元素只占用内部所包含内容使用的空间。如果不使用 CSS 的话，不可能将两个临近的行内级元素堆叠在一起。此外，如果某个元素属于行内级元素的话，当你将“margin-top/bottom”[①]或者“padding-top/bottom”[②]等格式应用到它上面去的时候，这些格式将会被忽略，只有水平方

① 分别表示顶部外边距和底部外边距。——译者注
② 分别表示顶部内边距和底部内边距。——译者注

向上（向左或者向右）的内边距和外边距设置才有效。图 1-2 给出了一个示例，用来说明当向本例中的 span 元素使用了 20 个像素的内边距之后，该元素在轮廓上所发生的变化。

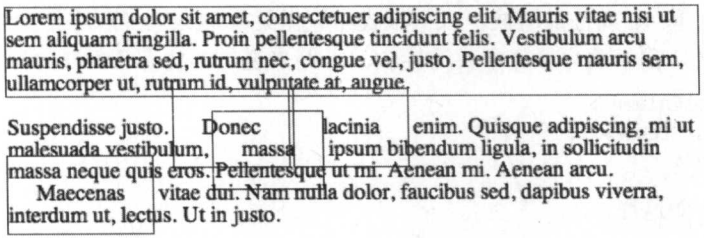


图 1-2 添加额外的内边距之后的行内元素

你可能已经看到，尽管“盒子”本身已经在 4 个方向上分别扩展了 20 个像素，但是顶部内边距和底部内边距并没有影响到周围的任何元素。

尽管通过运用 CSS 可以像行内元素那样显示块元素，或者像块元素那样显示行内元素，但是要注意，这并没有改变这些元素的意义，并且还是不能在 span 内部放置 div[①]。

关于块元素和行内元素之间的区别以及与它们相关的结构和操作，还有很多内容。要想进一步了解与这个主题有关的详细内容，推荐阅读由Tommy Olsson所写的文章*Block vs. Inline*（www.autisticcuckoo.net/archive.php?id=2005/01/11/block-vs-inline-1），这篇文章的内容非常精彩。此外，要想看看有关“盒子”内边距、外边距和边框格式效果的可视化演示，可以去看看Jon Hicks的3D CSS盒子模型（www.hicksdesign.co.uk/boxmodel）。

---

3. id和class属性

id 属性用来标识元素，并标示出网站的特定功能区，而 class 属性则用来对一个或者多个元素进行分类。当编写样式表或者脚本的时候，这些重要的属性将帮助定位相关的元素。我在整本书中都将引用这两个属性，但是现在只需要知道，每个页面中某个特定的 id 属性值只能使用一次，而 class 属性则可以使用多次（在同一个页面中，这些属性本身可以使用多次）。举例来说，如果文档的开头如下：

```
<body id="homepage">
```

那么在同一个页面的其他任何地方都不能将 id 的属性值设置为 homepage。但是如果文档开头如下：

```
<body class="homepage">
```

---

① 根据上下文，ins 或者 del 要么是块元素要么是行内元素。如果在其中一个元素中放置一个块元素，它们都将会成为块元素。但是如果将它们放到某个行内元素或者块元素内部，它们将成为行内元素。我将在下一章继续讨论这两个元素。

那么在同一个页面中就可以随便多少次将 class 的属性值设置为 homepage，但是要记住，它应用的始终还是同一个 CSS，而不管将其应用到什么标签上。

当使用 class 和 id 属性的时候，人们很容易根据元素的预期外观来给元素赋值，而不是根据元素本身的意义，我奉劝你最好不要这样做。举例来说，不要像下面这样赋值：

```
<div id="rightColumn">
<strong class="redText">
<p class="big">
```

而是应该像下面这样：

```
<div id="secondaryContent">
<strong class="important">
<p class="introduction">
```

为什么呢？这是因为某天你可能发现需要将元素设置为蓝色而不是红色，或者你希望将次要内容从右侧栏目移到左边。这样一来，class 和 id 的值就没有什么意义了。

注意：在XHTML中，id属性的开头不能是数字，所以类似于<body id="3columns">就是非法的，而<body id="columns3">则可行。

还可以将 id 和 class 同时应用到同一个元素上去：

```
<body id="homepage" class="page">
```

为了在 CSS 中引用这些属性值，写出相应的属性值后，在 id 的前面加上#号，在 class 前面加上点号（.），就像下面这样：

```
#homepage {
    background: blue;
}

.page {
    color: white;
}
```

这两个属性不会绑定到一个特定的标签，尽管可以将其中一个属性或者两个属性同时设置到这个标签上。

1.2　XHTML 和 HTML

在一般的 Web 项目中，到底使用 XHTML 还是 HTML，这通常并不是一个问题。因为今天的大多数 Web 设计师很自然地偏向于 XHTML，因为人们都认为它是属于新一代的高级技术，并且其中的“X”让它听起来非常“酷”。但实际上，XHTML 与 HTML 之间的差别并不像人们想

象的那么大。这一小节的目的就是想详细地讨论 XHTML 与早期版本的 HTML 之间到底有何不同，揭去 XHTML 和 HTML 的神秘面纱，消除人们对它们的误解。并对 MIME 类型背后存在的问题进行了调查，同时还讲述了在什么情况下使用 XHTML 优于 HTML，而又在什么情况下并非如此。

1.2.1 XHTML 和 HTML 之间的区别

有几个规则可以适用于 XHTML，但并不适用于 HTML。这些规则相当简单，你可能已经对它们中的一些（或者全部）有所了解，但是我还是要重申以下几点：

- 在 XHTML 中，<html>、<head>和<body>都是必需的标签。
- 必须设置<html>标签的 xmlns 属性，且其值为“http://www.w3.org/1999/xhtml”。
- 所有元素都必须结束。我在前面曾经提到这一点，但是只要记住任何起始标签要么有一个对应的结束标签（如果它是一个容器标签的话），要么是一个自结束元素“空格加斜线（space-plus-slash）”[①]。
- 所有标签都必须是小写。
- 任何属性值必须要么用单引号引起来，要么用双引号引起来。因而，class=page 就是不合法的，而 class="page"与 class='page'均是合法的。
- 所有属性必须有值。有些属性，比如<option>标签的 selected 属性，在 HTML 中可以使用简写形式，即<option selected>数据</option>，然而在 XHTML 中，必须这样编写：<option selected="selected">数据</option>。
- “&”符号必须编码。也就是说，应该将其写成“&amp”而不是仅仅写成“&”。不管是“&”在正文中还是在 URL 中，这一点始终都成立。

1.2.2 对 XHTML 和 HTML 的误解

当几年前XHTML脱颖而出的时候，它被很多人视为Web世界的“救世主”——它能够让人们脱离旧式的基于表格的HTML标记泥沼。由于XHTML正规性更强、规则更严格，人们期待XHTML能够更加易于编写、易于维护，并且能够在所有方面超越HTML。

实际上，除了前一节中所提到的一些小差别，XHTML 跟 HTML 比较起来并没有太大区别，要使用哪个版本更多地取决于如何使用它们编写代码。这一部分的内容将会展示你可能听过的坊间传说和一些误解，并向你揭示背后的真实情况。

1. XHTML的元素数目要比HTML多（少）

是的，可以说 XHTML 的元素数目比 HTML 多，也可以说 XHTML 的元素数目比 HTML 少，这取决于使用什么 doctype 来编写代码。如果只是拿 HTML 4.01 Strict 和 XHTML 1.0 Strict 比较，

① 即“ />”。——译者注

那么后者中的元素数目要比前者中的少。因为 HTML 4.01 Strict 中的已经不推荐使用的一些元素已经从 XHTML 1.0 Strict 中移除，它们是<dir>、<menu>、 、<isindex>、<applet>、<font>、<basefont>、<s>、<strike>、<u>、<iframe>和<noframes>。其中<iframe>（这个标签通常用来在页面中包含广告）可能是一个例外，除此之外，你不需要用到这些元素中的任何一个，因为它们已经有了更好的替代品——要么是更有意义的元素（举例来说，使用<del>来代替<s>和<strike>，将在下一章中更加详细讨论），要么是 CSS（举例来说，使用 CSS 字体属性来代替<font>元素）。于是，拿 HTML 4.01 Strict 跟 XHTML 1.0 Strict 做比较，答案就是 XHTML1.0 中的元素数目更少，但是因为所有这些元素在 HTML4.01 中已经不再推荐使用，所以在编码实践中不会产生任何实质上的差别。

如果doctype为Transitional，那么刚才提到的所有这些元素都是允许使用的，还有一些属性（比如target）可以用在<a>元素上。

如果你查看 XHTML 1.1 的话，你还会发现一些差别，它引入了 Ruby 元素[①]，这些元素通常用于东亚语种排版中。它去掉了 name 属性，用 xml:lang 来代替 lang 属性。XHTML 1.1 也必须使用 application/xhtml+xml 这样的 MIME 类型。

2. 与HTML相比，XHTML具有更好的错误检查功能（更严格或者更健壮）

这既可以说是正确的，也可以说是错误的。答案取决于你打算干什么。如果在 XHTML 页面中将 MIME 设置为 text/html，那么标记在健壮性方面不会超过 HTML，并且浏览器常常会试图纠正 XHTML 标记中存在的任何错误，并尝试着按照它们所猜测的意思显示。如果将 XHTML 页面的 MIME 类型设置为 application/xhtml+xml，那么页面中最细微的错误都会导致页面中止，而且通常只会显示一条 XML 解析错误消息。我将在本章后面详细介绍各种 MIME 类型。

3. 与HTML相比，XHTML的语义性更强或者结构性更强

不正确。前面曾经提到过，真正起作用的不是所用的技术，而是怎样使用技术。如果愿意的话，你可以编写出能够想象到的最糟糕的标记，其中混合了大量嵌套布局用的表格、换行标签，还有大量没有语义意义的元素，而这仍然还是一份合法的 XHMTL 文档。类似地，你也可以编写出以前从未见过的最纯粹的、最干净的、语义最丰富的页面，而它仍然可以用 HTML 4.01 编写出来。

4. 与HTML相比，XHTML更加简洁

不尽然。因为在有效的 XHTML 文档中，必须用引号将属性值引起来，每个元素必须有结束标签，页面头部还要有一大堆完整的标签和属性，因此在总体上 XHTML 页面实际上要比等价的 HTML 页面更“臃肿”。举例来说，Anne van Kesteren 的主页（http://annevankesteren.nl）开头部

---

① 更多信息请参见 www.w3.org/TR/ruby。